Failure Simulation in Software-Defined Networks with Differential Link Availability

Paulo Melo
Univ Coimbra, CeBER,
Faculty of Economics,
Coimbra, Portugal
pmelo@fe.uc.pt

Luisa Jorge
Research Centre in Digitalization
and Intelligent Robotics (CeDRI), IPB,
Bragança, Portugal
ljorge@ipb.pt

Teresa Gomes
University of Coimbra, INESC Coimbra,
Dept. of Electrical and Computer Engineering,
Coimbra, Portugal
teresa@deec.uc.pt

***Abstract*—In Software-Defined Networks (SDN) the placement of controllers is an important factor for overall network reliability. Whereas most studies assume that all links have similar link failure rates/availability, placing controllers taking into account differences in link availability can yield advantages. This is even more relevant if the network is designed with a particular subset of high availability links (a spine). After presenting integer linear programming formulations to support controller placement considering differential link availability, we propose a simulation approach to verify the reliability change due to that placement. From a simulational study using 2 networks, 30 different link availability configurations we found a definite advantage in overall reliability in using the knowledge of high availability links to place the controllers. However, while there were also advantages to using certain spines for individual networks, these advantages were not uniform, and varied according to networks, the number of controllers placed and link availability ratio. Further work is suggested to check whether fault patterns may influence the reliability advantage of individual spines.**

I. Introduction

In Software-Defined Networks (SDN) [8], the control function of data-plane components (such as switches and routers) is transferred to a control plane (controllers), so the network functions can be remotely programmed (application plane). However, regardless of how many controllers a particular network uses, each switching device must at all times be able to receive and act upon fine-grained commands given by one controller (of those assigned to it), which puts a premium on controller-switch connection (control-path) reliability. The correct placement of a suitable number of controllers is, therefore, a very important concern, and the so-called Controller Placement Problem (CPP) has been subject to a large number of studies (e.g. [3], [7]).

A. Motivation

To adequately place controllers to provide adequate network reliability, the concept of link availability is relevant. In fact, the availability of the control path is proportional to the product of link availability of the links comprising it and, all things being equal, selecting links with higher availability would be expected to increase overall control path availability. Networks availability concerns are frequently driven by the reliability needs of the most demanding network services and as such most networks do not assume differences in link availability on the design or CPP stage. Such differences may occur, either by network equipment evolution and degradation or by explicit design (like proposed in [1]) and should be considered when placing controllers. Analytical methods have been the preferred method to assess placement quality, but in this work we propose an alternative mechanism which intends to actually measure network reliability using simulation. While acknowledging the disadvantages in time and processing requirements, we propose the added flexibility may prove useful in addressing some less structured networks and fault patterns.

B. Related Work and Contribution

While the problem of optimal placement of SDN controllers (CPP) has been widely studied (for some recent reviews, see [3], [7], [12]) no definitive answer is yet evident. The problem of defining what is optimal is itself complex, since many conflicting objectives may be present, usually different specializations of latency, reliability and cost [9], but which may also include less common objectives like energy-efficiency or information sharing requirements [3]. Furthermore, many of these objectives will be in conflict, hence the requirement for multi-objective optimization also being common (see for instance the multiple variations of the Pareto-based Optimal COntroller placement (POCO) framework [6]). Regardless of the actual mechanism used, however, the objective of CPP usually includes the definition of the number and placement of SDN controllers for a particular network.

Our work focuses in reliable control-paths. While the mechanisms of failure on an SDN network are quite varied (see e.g. [15]) this work will explore link failures on the data plane which translate to link failures in the control plane for the control connection between controller and controlled switch (in terms of reliability, we are concerned in supporting multiple controllers and multiple control paths, but in this work we aren't concerned with inter-controller traffic). Among the many proposals to solve this problem, our work is most closely related to the proposals of Vizarreta et al. [14], particularly in the support for multiple control paths to ensure control plane reliability, but includes support for taking explicitly into consideration the availability of individual links when deciding which ones to consider. While a recent work [11] also takes into account spine information for SDN networks, our proposal

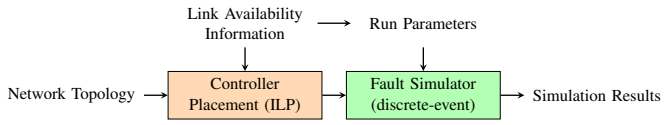

Fig. 1. Simulation Process

is not to design reliable SDN networks containing spines, but rather to allow evaluation of their characteristics.

The usage of discrete-event simulation in SDN is common, particularly to evaluate the performance of particular protocols or network architectures [5]. However, reliability studies frequently use it in the context of Monte-Carlo methods [4], unlike in the discrete-event simulation used in this work.

The main contributions of this paper are:

- Integer Linear Programming (ILP) formulations for CPP considering link availability, for single and multiple control paths and for a single or for a pair of controllers per switch;
- The structure of a simulational approach to verify the placement (see Figure 1). This process can be applied to multiple placement approaches, since it isolates the controller placement stage from the subsequent fault simulation stage;
- The design and implementation of an expandable simulation artifact applying such approach to verify the reliability results of particular controller placements in networks with differential link availability.

II. Simulational Approach

To test the approach proposed, we generated (adapting minimally the formulations presented in [14]) a set of controller placement ILP models which try to generate controller placements taking into account different availability levels in edges. After this, we created a discrete event fault simulator to verify whether this placement would yield better results than a placement not using this information.

A. Controller placement strategies in differential link availability networks using linear programming

TABLE I
Basic Notation

Symbol	***Definition***
V	set of switches
E	set of physical links (i,j) with $i,j \in V$
$n_V = \|V\|$	number of switches
k	number of controllers
d_{ij}	penalty associated with link (i,j)
ρ_{ij}	availability value for link (i,j)

Using the notation defined in Table I, we define $F^n \subset E$ and $S^n \subset E$ as the set of edges of the first and second control-paths, respectively, to a controlled switch $n \in V$. We assume the paths to be node-disjoint (and edge-disjoint), therefore $F^n \cap S^n = \varnothing$.

We can then define two sets of binary variables:

$$U = \left\{ u_{ij}^n \in \{0,1\} \,|n \in V, (i,j) \in E \right\}$$
$$W = \left\{ w_{ij}^n \in \{0,1\} \,|n \in V, (i,j) \in E \right\}$$

Where:

$$u_{ij}^n = \begin{cases} 1 \iff (i,j) \in F^n \\ 0 \iff (i,j) \notin F^n \end{cases}$$
$$w_{ij}^n = \begin{cases} 1 \iff (i,j) \in S^n \\ 0 \iff (i,j) \notin S^n \end{cases}$$

To solve the problem, we add additional binary variable sets X and Y:

$$X = \left\{ x_i^j \in \{0,1\} \,|i \in V \wedge j \in V \right\}$$
$$Y = \left\{ y^j \in \{0,1\} \,|j \in V \right\}$$

Where:

$$x_i^j = \begin{cases} 1 \iff i \text{ is controlled by } j \\ 0 \iff i \text{ is not controlled by } j \end{cases}$$
$$y^j = \begin{cases} 1 \iff j \text{ has a controller placed there} \\ 0 \iff j \text{ does not have a controller placed there} \end{cases}$$

To simplify the flow control equations, we can also define an auxiliary constant (adapted from [14]):

$$t_{nm} = \begin{cases} -1 \iff n = m \\ 0 \iff n \neq m \end{cases}$$

Using this notation, the controller placement problem for k controllers, can be formulated as a integer (binary) linear programming problem intending to obtain F^n as (for conciseness we omit the explanations present in [14]):

$$\text{Minimize } Z = \sum_{n \in V} \sum_{(i,j) \in E} d_{ij} u_{ij}^n$$

Such that:

$$\sum_{j \in V} y^j \leq k \tag{1}$$

$$\sum_{j \in V} x_i^j = 1 \qquad \forall i \in V \tag{2}$$

$$\sum_{i \in V} x_i^j \leq n_V y^j \qquad \forall j \in V \tag{3}$$

$$\sum_{(i,m) \in E} u_{im}^n - \sum_{(m,j) \in E} u_{mj}^n - x_m^n = t_{nm} \quad \forall n, m \in V \tag{4}$$

If one assumes uniform costs for all edges ($d_{ij} = 1, \forall (i,j) \in E$) the problem will provide paths for hop-count minimization, while with different link costs we can incorporate other objectives. If however some links have different availability characteristics, the solution to this problem may not yield a sufficiently robust controller placement. A simple solution to prevent the usage of lower availability links is simply to add additional constraints/variable bounds to exclude links with lower than the desired link minimum availability level P:

$$u_{ij}^n \leq 0 \qquad \forall (i,j) \in E | \rho_{ij} < P \tag{5}$$

1) Disjoint paths for the same controller: As shown in [14], the previous problem statement may be expanded to support multiple control paths for the same controller (RCP-DCP problem), by expanding the ILP problem with variables for a second path for the same controller (defined by set S^n). We assume the paths to be edge-disjoint, therefore $F^n \cap S^n = \varnothing$. We also add restrictions to the problem for node disjunction (except for the controller and the node being controlled on each path).

To try to minimize the length of the first path before the length of the second, the objective function was amended to increase the weight placed on U versus W.

$$\text{Minimize } Z = \sum_{c \in V} \sum_{(i,j) \in E} d_{ij}(n_V u_{ij}^n + w_{ij}^n) \tag{6}$$

To the restrictions defined in equations (1)–(4) we add new ones to support the second path flow control

$$\sum_{(i,m) \in E} w_{im}^n - \sum_{(m,j) \in E} w_{mj}^n - x_m^n = t_{nm} \quad \forall n, m \in V \tag{7}$$

and some restrictions to guarantee node disjunction:

$$\sum_{(m,j) \in E} (u_{mj}^n + w_{mj}^n) \leq 1 - t_{nm} \qquad \forall n, m \in V \tag{8}$$

$$\sum_{(i,m) \in E} (u_{im}^n + w_{im}^n) - x_m^n \leq 1 \qquad \forall n, m \in V \tag{9}$$

$$\sum_{(i,m) \in E} (u_{im}^n + w_{im}^n) \leq 1 \qquad \forall n, m \in V \tag{10}$$

As stated previously, the inclusion of the restrictions of equation (5) will force the first path to be on high availability links. Equation (10) is redundant for this problem, however it is defined here since it will be used in all multiple-controller problems.

2) Disjoint paths to different controllers: A variation of the previous model allows the paths to be directed to different controller replicas. Almost the same notation can be reused, although we now define $F_{c_1}^n \subset E$ as the set of edges in the first control-path from controller $c_1 \in V$ to switch $n \in V$, whereas $S_{c_2}^n \subset E$ is the set of edges belonging to a second path from controller $c_2 \in V$ to $n \in V$. We continue to assume the paths to be edge-disjoint and node-disjoint (except for the controlled node), therefore $F_{c_1}^n \cap S_{c_2}^n = \varnothing$.

As in [14], we redefine the set X to correspond to the first controller, and must create a new set of variables Z which perform the same task for the second controller:

$$X = \left\{ x_i^j \in \{0,1\} \,|\, i \in V \wedge j \in V \right\}$$
$$Z = \left\{ z_i^j \in \{0,1\} \,|\, i \in V \wedge j \in V \right\}$$

$$x_i^j = \begin{cases} 1 \iff i \text{ is controlled by } c_1 \text{ placed in } j \\ 0 \iff i \text{ is not controlled by } c_1 \text{ placed in } j \end{cases}$$
$$z_i^j = \begin{cases} 1 \iff i \text{ is controlled by } c_2 \text{ placed in } j \\ 0 \iff i \text{ is not controlled by } c_2 \text{ placed in } j \end{cases}$$

The ILP formulation uses the same objective function as before (equation (6)) and also the constraints in equations (1)–(4), but some additional constraints must be added:

$$\sum_{i \in V} z_i^j \leq n_V y^j \qquad \forall j \in V \tag{11}$$

The constraint group in equation (7) must be adapted to point to the new controller:

$$\sum_{(i,m) \in E} w_{im}^n - \sum_{(m,j) \in E} w_{mj}^n - z_m^n = t_{nm} \quad \forall n, m \in V \tag{12}$$

Finally, some new disjunction constraints must be added to the ones in equations (8) and (10):

$$x_i^j + z_i^j \leq 1 \qquad \forall i, j \in V \tag{13}$$

As before, the inclusion of the restrictions of equation (5) will force the first path to be placed on high availability links.

B. Failure simulation using OMNeT++

Once the controller placement program has been solved, we can now look into any actual reliability gains achieved by taking explicitly into account the different availability of network links when determining the controller placement. To do so, we developed a failure simulator, using the discrete event simulation platform OMNeT++ [13].

1) A rationale for failure simulation: While reliability in networks has been studied frequently using analytical models [2], [4], the usage of simulation for its study can be justified frequently due to its inherent flexibility, allowing the creation of models with support for various customization capabilities which would be harder to model with simple analytic models.

Some of these include multiple networks and edge characteristics, including multiple reliability levels and non-linear reliability for some edges. Also the multiple fault models can be successfully integrated, including changes on failure rate and duration, sequential failures and "geographically" correlated failures. While it can be argued that specific analytic models can be developed to support each of these characteristics, it would be hard to create a general analytic model able to encompass all of these.

2) Simulator structure: An application was developed in OMNeT++ able to load a network, annotated with reliability information (a set of links with a different availability rating) and a set of SDN network controllers.

It uses the base simulation support, and is created via four custom built simple modules (see Figure 2):

Loader responsible for loading the network with reliability annotations, and an additional file containing the controller placement, and create an OMNeT++ network layer model, comprised of nodes and links to visually represent the information;

DataBase supports the internal network model and acts as a focal point to exchange information regarding link faults

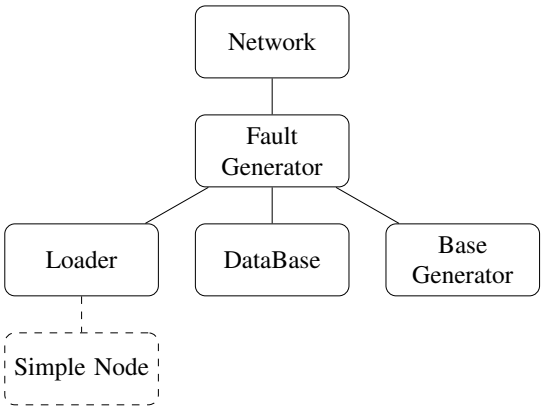

Fig. 2. Simulator Main Components

TABLE II
SIMULATOR INPUT VARIABLES

Variable	*Module*	*Description*
nodeModType	Loader	The module type to associate to a node (to support node specialization)
simFilePath	DataBase	Simulation files path (common to all files on a run)
simFileName	Loader	Simulation file name
ratio_spine	Base Generator	Fault probability ratio between high-availability and low-availability links
generation_delay	Base Generator	Interval between generated faults
failure_duration	Base Generator	Duration of each fault

and path failures, and updates (as needed) the OMNeT++ network layer;

Base Generator generates faults, communicating with the DataBase to update the internal network information;

Simple Node is used to represent each switch in the SDN network (each instance of simple node is procedurally created and linked by the loader module).

These simple modules are in turn integrated in the complex module *Fault Generator*, which, as required by OMNeT++, is integrated in a *Network* for actual simulation.

This simulation structure allows for redefining the different components (e.g. specializing the *Base Generator* with another one that allows for different kinds of faults) to support evolution. Isolating the *Fault Generator* from the network loader (*Loader*) allows for supporting several networks and even new network formats without a needing to greatly change the simulator (or at all). To support it, a format for loading has been specified (Sim File format) and a file converter has been developed to create it from the network data (in SDNlib [10] format) and controller placement information.

A set of input variables is defined in the simulator, to allow for run configuration (see Table II). Besides the input variables, additional customization may be achieved via the network file generator (e.g. allowing for different reliability profiles on the same network, and different number of controllers) outside of the simulator.

The set of output variables is comprised by the disconnected time fraction (per-switch, per-link and per-network) for each switch in terms of each controller-switch path (and when applicable, of all the controllers assigned to a switch – which would imply switch disconnection). Further output variables can be added (by specializing the DataBase module).

TABLE III
SIMULATOR RUN PARAMETERS

Variable	*Unit type*	*Values*	*#*
ratio_spine	ratio	$1, 4, 7, 10, 13$	5
generation_delay	second	$10, 15, 20, 25$	4
failure_duration	second	$5, 10, 15, 20, 25, 30$	6
repetition	seed	$0, 1, 2, 3, 4$	5

III. RESULTS AND DISCUSSION

The simulator was used with two SNDlib [10] networks, Germany50 (50 nodes and 88 edges), and Polska (15 nodes and 18 edges) topologies, with a set of 20 different high-availability spanning trees/spines [1] plus one with uniform availability for all links for the Germany50 network, and 9 different spines plus one with uniform availability for all links the Polska network.

For each of these network/spine pairs, solutions to the controller placement problem were created for a single path for the controller, for disjoint paths to a single controller per node, and for disjoint paths to different controllers, with unit link costs. The problems were solved (using CPLEX) for a total number of controllers k ranging from 1 to 9 (which is excessive for Polska, but was used for regularity). For 4 of the spines in the Germany50 network, with a single controller it was not possible to find disjoint paths to that controller if the first path would need to use exclusively spine links. For all the remaining, a solution was possible for all the options. All the placements were run twice, once using unit link costs (hop-count) and the other using geographical distance costs.

For the solutions thus created, faults were simulated taking into account the differential availability of the spine links. The options selected are presented in Table III, which meant that each tuple (network/spine/placement) was run 600 times. Notice that the values of delay between faults and duration of those faults are intentionally kept small and near each other. These values were selected so that simultaneous failures could be possible without having to run rare-event simulations (for which OMNeT++ is not best optimized). However, even if that made the failure rates inordinately high, it should not distort too much the insights gained from the simulation.

While the results are too many to present here in full, an aggregate analysis allow us to state some preliminary findings, which will be illustrated with some graphs for a subset of the results (focusing only in simultaneous failures/switch disconnection):

- When there is a spine in the network, the probability of disconnection by faults is always lower if the controllers are placed taking it into account, for $ratio\_spine \geq 4$ (ratio between the availability rating of the edges of the

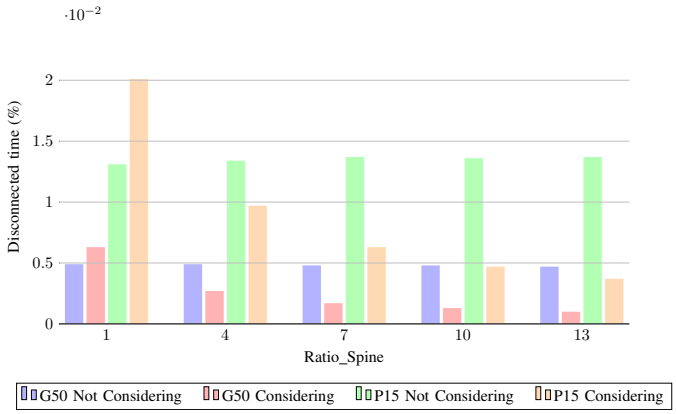

Fig. 3. Disconnection percentage when considering or not the differential availability in the network when placing controllers – averages for Germany50 (G50) and Polska (P15) networks

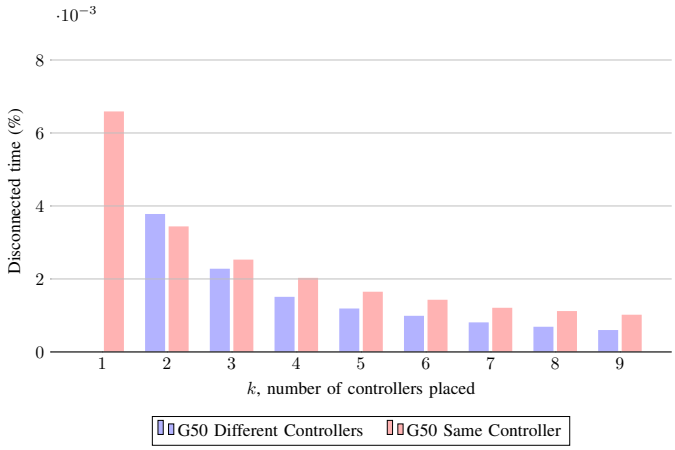

Fig. 4. Disconnection per controller number – disjoint paths versus separate controllers – Germany50 average

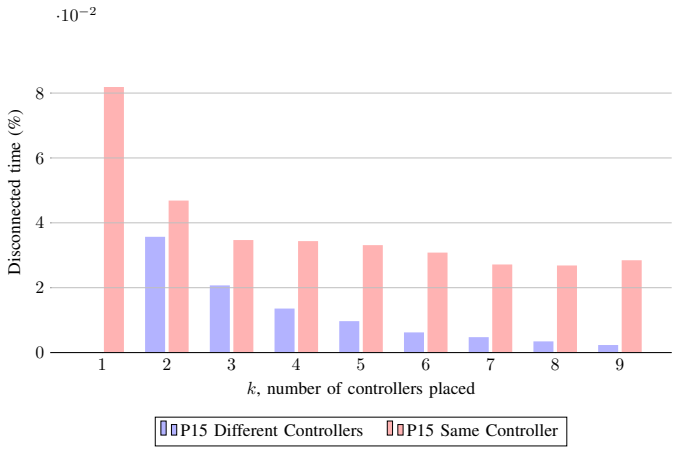

Fig. 5. Disconnection per controller number – disjoint paths versus separate controllers – Polska average

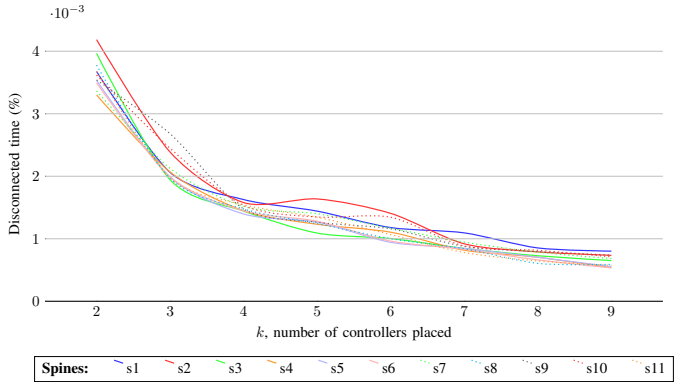

Fig. 6. Reliability when using different spines – Germany50 average, $k = 2$ to 9

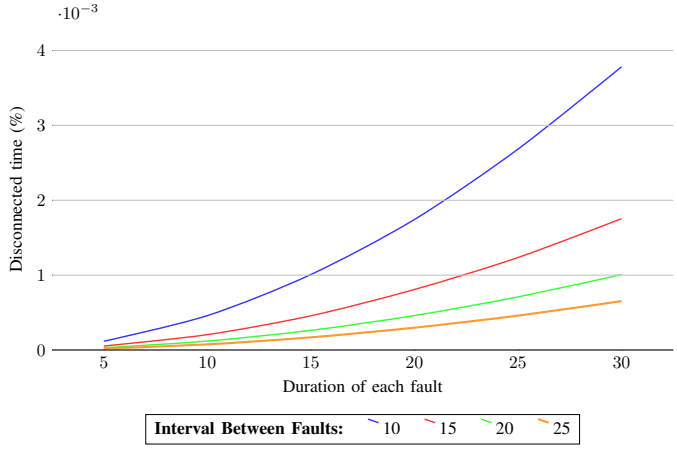

Fig. 7. Reliability vs Failure Duration and Frequency – Germany50 average

spine and the others), as seen in Figure 3. This advantage further increases for higher values of ratio_spine. Conversely, as expected, when forcing one path on links with no availability advantage ($ratio\_spine = 1$) there is increased disconnection.

- In general, the disconnection rate is lower when using two separate controllers than when using two disjoint paths to the same controller (but there is a marginal exception in the Germany50 network when using only 2 controllers – see Figure 4).
- More controllers on the network generally imply less chance of disconnection due to faults (a possible exception occurs when the number of controllers is very large on the Polska network, suggesting a tipping point where too many controllers on a small network actually decrease reliability – see Figure 5).
- There is some impact on the choice of specific high-availability links (that is, the usage of particular spines), but this varies strongly with the number of controllers, so no pattern can be discerned *a-priori* (see Figure 6 which shows a subset of the Germany-50 spines).
- As expected, the probability of disconnection upon fault

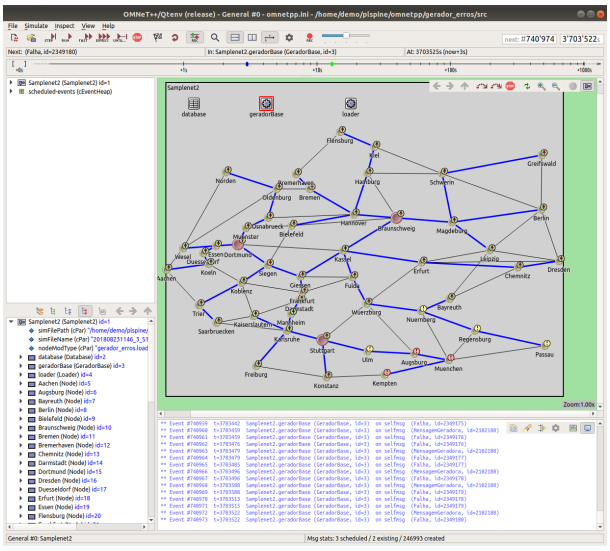

Fig. 8. Simulator Graphical User Interface

always increases with increasing duration and frequency of faults (see Figure 7).

While most simulations were run without the graphical interface, a frontend to the simulator was developed using the built-in capabilities of OMNeT++ (see Figure 8). In this picture you can see the higher availability links (blue links). It is also visible that in this simulation we have placed 3 SDN controller nodes (Dortmund, Braunschweig e Stuttgart). Finally it also shows that a fault (not directly displayed) is affecting the path from Augsburg, Kemptem and Muenchen to their main controller, and the path from Nuernberg, Passau, Regensburg and Ulm to their secondary controller (no failed nodes were created by this single fault).

IV. Conclusions and Future Work

We presented a simulational approach to validate the placement of controllers in a SDN network with different availability links. The results show that taking the differential availability in consideration is useful, particularly for large availability differences among links. However the advantage is influenced by the actual high-availability links present, and no *a priori* relation was discernible from the results (and neither traffic displacement nor scalability effects were considered in the analysis). Further work is required in this area, and is also suggested in the area of complex fault and reliability patterns, for which the flexibility granted by the approach is ideally suited.

Acknowledgment

This work has been partially funded by ERDF Funds through the Centre's Regional Operational Program and by National Funds through the FCT - Fundação para a Ciência e a Tecnologia, I.P. under the project CENTRO-01-0145-FEDER-029312, and also by FCT Grants UIDB/00308/2020 (INESC Coimbra), UIDB/05757/2020 (CeDRI), and UIDB/05037/2020 (CeBER). The authors would like to thank Prof. Rita Girão-Silva and Prof. Lúcia Martins for their contribution in spine generation and Prof. Amaro de Sousa for his comments and discussion.

References

[1] A. Alashaikh, T. Gomes, and D. Tipper, "The Spine concept for improving network availability," *Computer Networks*, vol. 82, pp. 4–19, 5 2015. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1389128615000687

[2] S. K. Chaturvedi, *Network Reliability*. Hoboken, NJ, USA: John Wiley & Sons, Inc., 5 2016. [Online]. Available: http://doi.wiley.com/10.1002/9781119224006

[3] T. Das, V. Sridharan, and M. Gurusamy, "A Survey on Controller Placement in SDN," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 472–503, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/8802245/

[4] I. B. Gertsbakh and Y. Shpungin, *Models of Network Reliability*. CRC Press, 4 2016. [Online]. Available: https://www.taylorfrancis.com/books/9781439817421

[5] N. Gray, T. Zinner, S. Gebert, and P. Tran-Gia, "Simulation Framework for Distributed SDN-Controller Architectures in OMNeT++," in *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST*. Springer Verlag, 10 2017, vol. 191, pp. 3–18. [Online]. Available: http://link.springer.com/10.1007/978-3-319-52712-3_1

[6] D. Hock, S. Gebert, M. Hartmann, T. Zinner, and P. Tran-Gia, "POCO-framework for Pareto-optimal resilient controller placement in SDN-based core networks," in *2014 IEEE Network Operations and Management Symposium (NOMS)*. IEEE, 5 2014, pp. 1–2. [Online]. Available: http://ieeexplore.ieee.org/document/6838275/

[7] B. Isong, R. R. S. Molose, A. M. Abu-Mahfouz, and N. Dladlu, "Comprehensive Review of SDN Controller Placement Strategies," *IEEE Access*, vol. 8, pp. 170 070–170 092, 9 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9195810/

[8] D. Kreutz, F. M. V. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 1 2015. [Online]. Available: http://ieeexplore.ieee.org/document/6994333/

[9] J. Lu, Z. Zhang, T. Hu, P. Yi, and J. Lan, "A Survey of Controller Placement Problem in Software-Defined Networking," *IEEE Access*, vol. 7, pp. 24 290–24 307, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8618449/

[10] S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly, "SNDlib 1.0 - Survivable Network Design Library," in *Proceedings of the 3rd International Network Optimization Conference (INOC 2007), Spa, Belgium*, 2007. [Online]. Available: http://www.zib.de/orlowski/Paper/OrlowskiPioroTomaszewskiWessaely2007-SNDlib-INOC.pdf.gz

[11] D. Santos, T. Gomes, and D. Tipper, "Software-Defined Network Design driven by Availability Requirements," in *2020 16th International Conference on the Design of Reliable Communication Networks DRCN 2020*. IEEE, 3 2020, pp. 1–7. [Online]. Available: https://ieeexplore.ieee.org/document/9089377/

[12] A. K. Singh and S. Srivastava, "A survey and classification of controller placement problem in SDN," *International Journal of Network Management*, vol. 28, no. 3, p. e2018, 5 2018. [Online]. Available: http://doi.wiley.com/10.1002/nem.2018

[13] A. Varga, "OMNeT++," in *Modeling and Tools for Network Simulation*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 35–59. [Online]. Available: http://link.springer.com/10.1007/978-3-642-12331-3_3

[14] P. Vizarreta, C. M. Machuca, and W. Kellerer, "Controller placement strategies for a resilient SDN control plane," in *2016 8th International Workshop on Resilient Networks Design and Modeling (RNDM)*. IEEE, 9 2016, pp. 253–259. [Online]. Available: http://ieeexplore.ieee.org/document/7608295/

[15] Y. Yu, X. Li, X. Leng, L. Song, K. Bu, Y. Chen, J. Yang, L. Zhang, K. Cheng, and X. Xiao, "Fault Management in Software-Defined Networking: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 349–392, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8456508/